Internet History

EE122 Fall 2012

Scott Shenker

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley

Administrivia

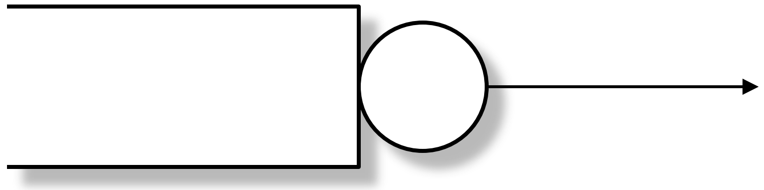
- Keep those questions coming!
  - But will take a while for me to calibrate length of lectures
- Questions about switching sections?
  - Email Kay keo@eecs.berkeley.edu, cc me.
- Check your bspace email address!
  - Make sure it is an account you check
  - If you miss a future bspace message, it is ***<u>your</u>*** fault
- Everyone should be on Piazza now
  - Again, we now view Piazza communications as reliable

Outline for Today

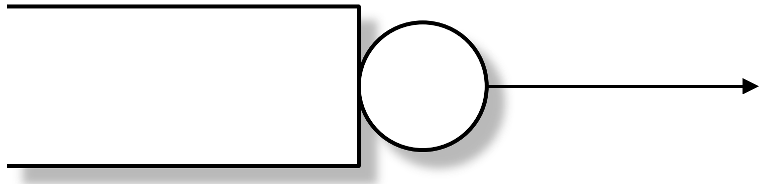
- Finishing up queueing
- Statistical Multiplexing
- Why did we choose packet-switching?
- Internet history
- Internet design goals
- Protocols, Clients and Servers,….

Finishing Up Queueing Delays

Smooth Arrivals = No Queueing Delays



Bursty Arrivals = Queueing Delays



There is substantial queueing delay even though link is underutilized

Queueing Delay Review

- Does not happen if packets are evenly spaced
  - And arrival rate is less than service rate
- Queueing delay caused by “packet interference”
  - Burstiness of arrival schedule
  - Variations in packet lengths
- Made worse at high load
  - Less “idle time” to absorb bursts
  - Think about traffic jams in rush hour….

Jitter

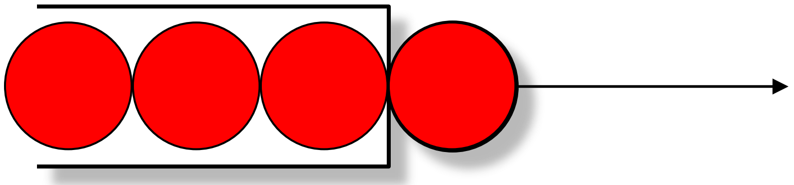
- Difference between minimum and maximal delay
- Latency (propagation delay) plays no role in jitter
  - Nor does transmission delay for same sized packets
- Jitter typically just differences in queueing delay
- Why might an application care about jitter?

Packet Losses

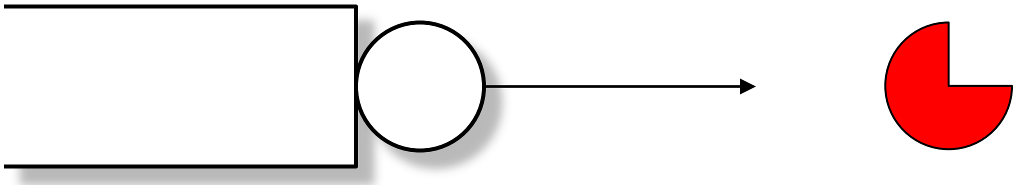
- Packets can be “dropped” or lost

- How?

Packet Losses: Buffers Full



Packet Losses: Corruption



Packet Losses

- Where should packet corruption be detected?
  - In network?
  - At host?

- Other ways packets can be lost?
  - TTL!

Basic Queueing Theory Terminology

- Arrival process: how packets arrive
  - Average rate A
  - Peak rate P
- Service process: transmission times
  - Average transmission time
  - For networks, function of packet size
- W: average time packets wait in the queue
  - W for “waiting time”
- L: average number of packets waiting in the queue
  - L for “length of queue”
- Two different quantities

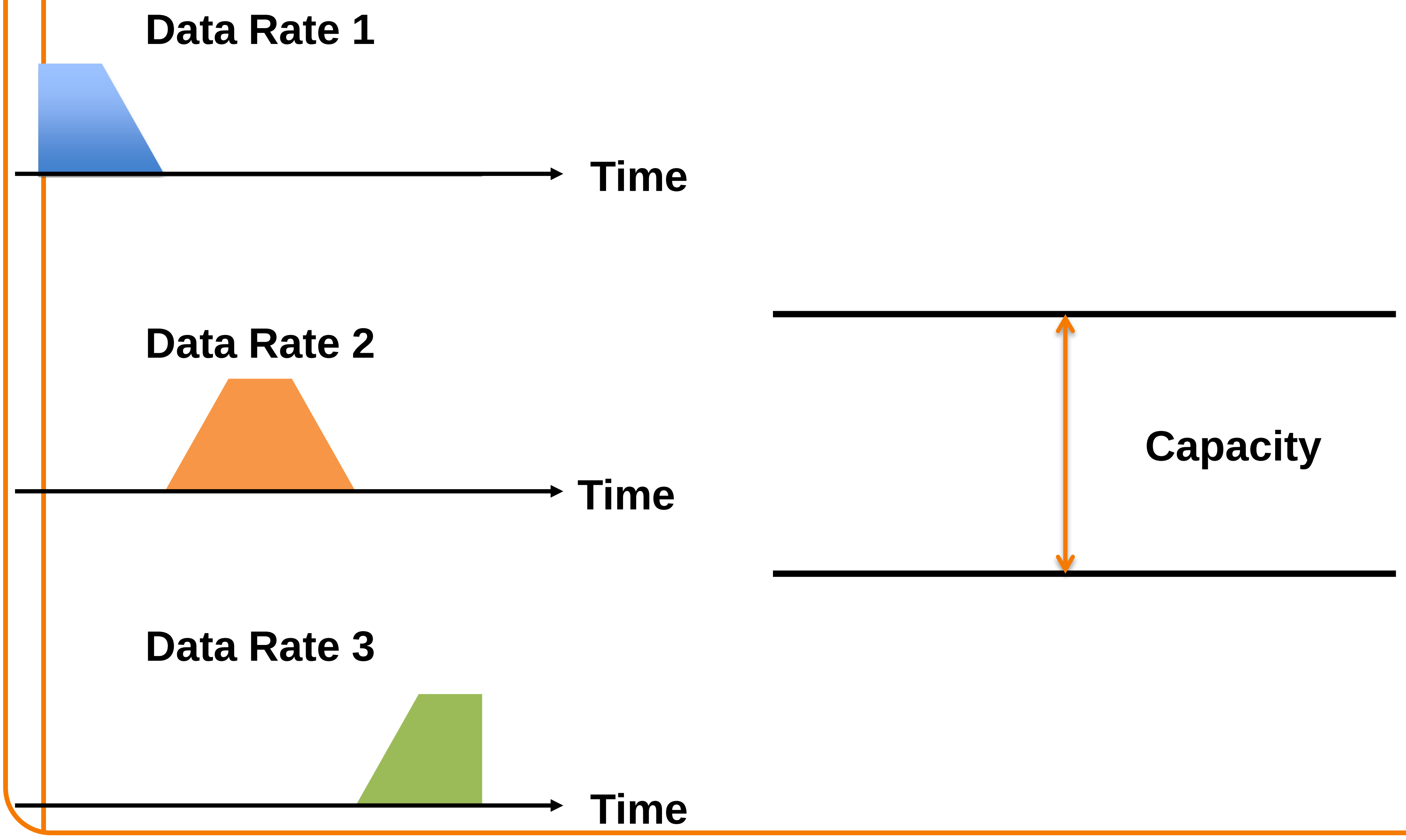
Little's Law (1961)

$$L = A \times W$$

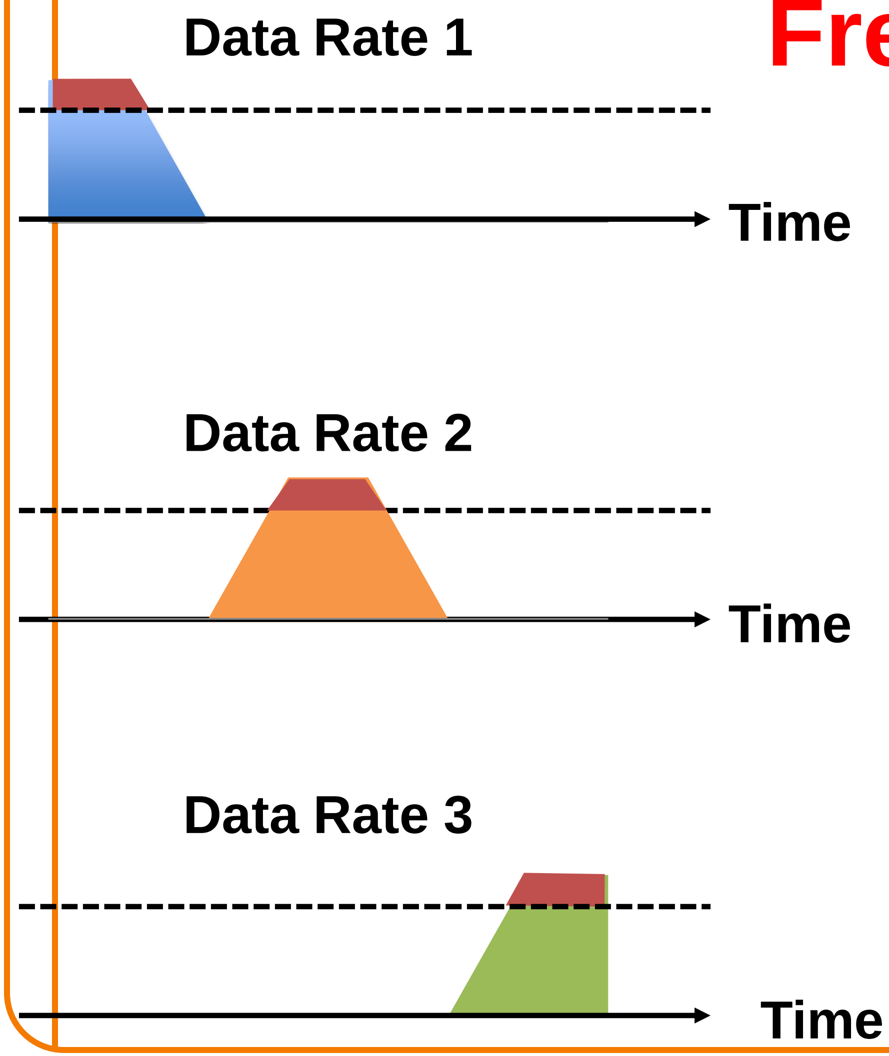
- Compute L: count packets in queue every second
  - This is the straightforward approach, easy to compute
- How often does a packet get counted? W times
  - The average arrival rate determines how frequently this total queue occupancy should be added to the total
- Why do you care?
  - Easy to compute L, harder to compute W
  - But W is what applications notice, so that's what we want

Statistical Multiplexing

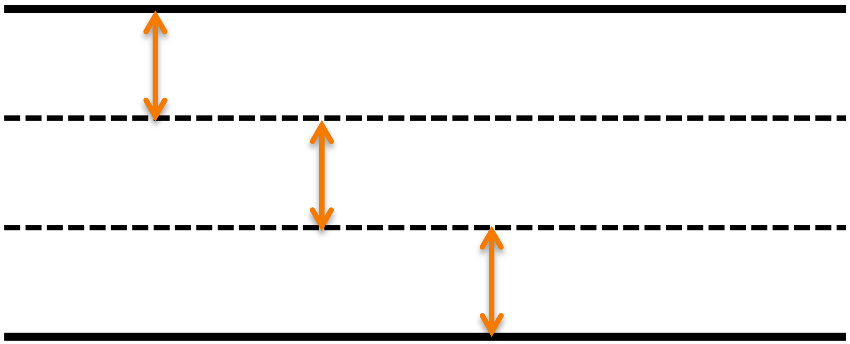
Three Flows with Bursty Arrivals



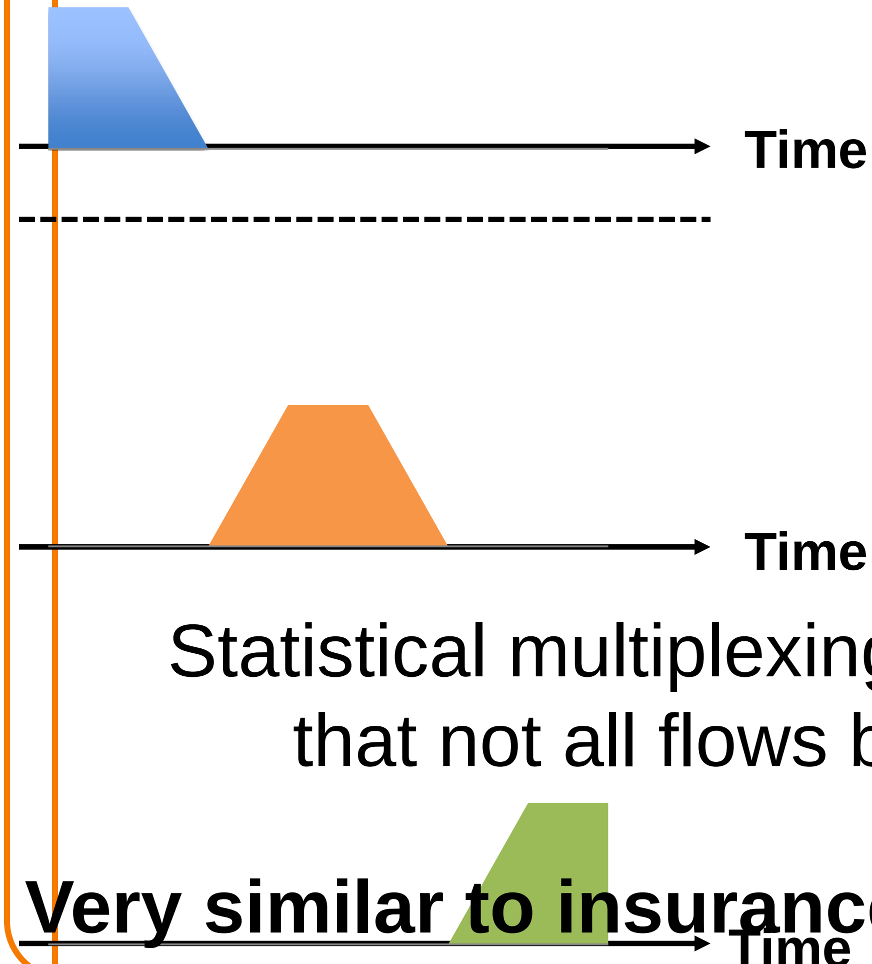
When Each Flow Gets 1/3rd of Capacity



Frequent Overloading



When Flows Share Total Capacity



No Overloading

Statistical multiplexing relies on the assumption that not all flows burst at the same time.

Very similar to insurance, and has same failure case

Improved Delays: M/M/1 Queue

- Consider n flows sharing a single queue
- Flow: random (Poisson) arrivals at rate λ
- Random (Exponential) service with average $1/\mu$
- Utilization factor: $\rho = n\lambda/\mu$
  - If $\rho > 1$, system is unstable
- Case 1: Flows share bandwidth
  - Delay = $1/(\mu - n\lambda)$
- Case 2: Flows each have $1/n^{th}$ share of bandwidth
  - No sharing
  - Delay = $n/(\mu - n\lambda)$ **Not sharing increases delay by n**

If you still don't understand stat mux

- Will cover in section….
- Will not be tested on M/M/1 content
  - This is just for "fun"

Recurrent theme in computer science

- Greater efficiency through “sharing”
  - Statistical multiplexing
- Phone network rather than dedicated lines
  - Ancient history
- Packet switching rather than circuits
  - Last lecture
- Cloud computing
  - Shared datacenters, rather than single PCs

Choosing a Network Design

If you were building a network….

- Which would you choose?
  - Circuit switched?
  - Packet-switched (**Datagram**)?

- Let's review the strengths and weaknesses

Advantages of Circuit Switching

- **Guaranteed bandwidth**
  - Predictable communication performance
- **Simple abstraction**
  - Reliable communication channel between hosts
  - No worries about lost or out-of-order packets
- **Simple forwarding**
  - Forwarding based on time slot or frequency
  - No need to inspect a packet header

Disadvantages of Circuit-Switching

- **Wasted bandwidth**
  - Bursty traffic leads to idle connection during silent period
- **Blocked connections**
  - Connection refused when resources are not sufficient
  - Unable to offer “okay” service to everybody
- **Network state**
  - Network nodes must store per-connection information
  - This makes failures more disruptive!

Disadvantages of Datagram

- **Worse service for flows**
  - No promises made about delays, just "**best effort**"
  - Packets might be dropped or delivered out of order
  - End hosts must cope with out-of-order packets
- **Must deal with congestion**
  - Without reserved resources, must cope with overloads
  - Overloads can result in bad service for flows
- **More complicated forwarding**
  - Per-packet lookups, etc.

Advantages of Datagram

- **Recovery from failures**
  - Routers don’t know about individual flows, so it is easy to fail over to a different path
- **Efficiency**
  - Exploits of statistical multiplexing
- **Deployability**
  - Easier for different parties to link their networks together because they are not promising to reserve resources for one another

Choosing a Design for the Internet

- Which would you choose? And why?

The paradox of the Internet's design

- As we will discuss later, one of the main design goals is to support a wide range of apps
- These applications have different requirements
- Shouldn't the Internet support them all?

Diversity of application requirements

- Size of transfers
- Bidirectionality (or not)
- Delay sensitive (or not)
- Tolerance of jitter (or not)
- Tolerance of packet drop (or not)
- Need for reliability (or not)
- Multipoint (or not)
- …..

Diversity of application requirements

- Size of transfers
- Bidirectionality (or not)
- **Delay sensitive** (or not)
- **Tolerance of jitter** (or not)
- **Tolerance of packet drop** (or not)
- Need for reliability (or not)
- Multipoint (or not)
- …..

What service should Internet support?

- Strict delay bounds?
  - Some applications require them
- Guaranteed delivery?
  - Some applications are sensitive to packet drops
- **No applications mind getting good service**
  - Why not require Internet support these guarantees?

Important life lessons

- People (applications) don't always need what they think they need
- People (applications) don't always need what we think they need
- Flexibility often more important than performance
  - **But typically only in hindsight!**
  - Example: cell phones vs landlines
- *Architect for flexibility, engineer for performance*

Applying lessons to Internet

- Requiring performance guarantees would limit variety of networks that could attach to Internet
- Many applications don't need these guarantees
- And those that do?
  - Well, they don't either (usually)
  - Tremendous ability to mask drops, delays
- And ISPs can work hard to deliver good service without changing the architecture (engineering)
- If the Internet had focused on voice applications early, it might have made different choices

Bottom Line

- The choice of datagram service is not as obvious as it might seem today
- Great vision on the part of the Internet designers
- And lucky that they were focused on applications that did not need great service

5 Minute Break

Questions Before We Proceed?

Internet History

Timeline

1961 Baran and Kleinrock advocate packet switching

1962 Licklider's vision of Galactic Network

1965 Roberts connects two computers via phone

1967 Roberts publishes vision of ARPANET

1969 BBN installs first IMP at UCLA

IMP: Interface Message Processor

1971 Network Control Program (protocol)

1972 Public demonstration of ARPANET

The beginning of the Internet revolution

- Kleinrock's group at UCLA tried to log on to Stanford computer: His recollection of the event…
- We typed the L…
  - "Do you see the L?"
  - "Yes, we see the L."
- We typed the O…
  - "Do you see the O?"
  - "Yes, we see the O."
- Then we typed the G…
  - …and the system crashed!

Timeline continued…

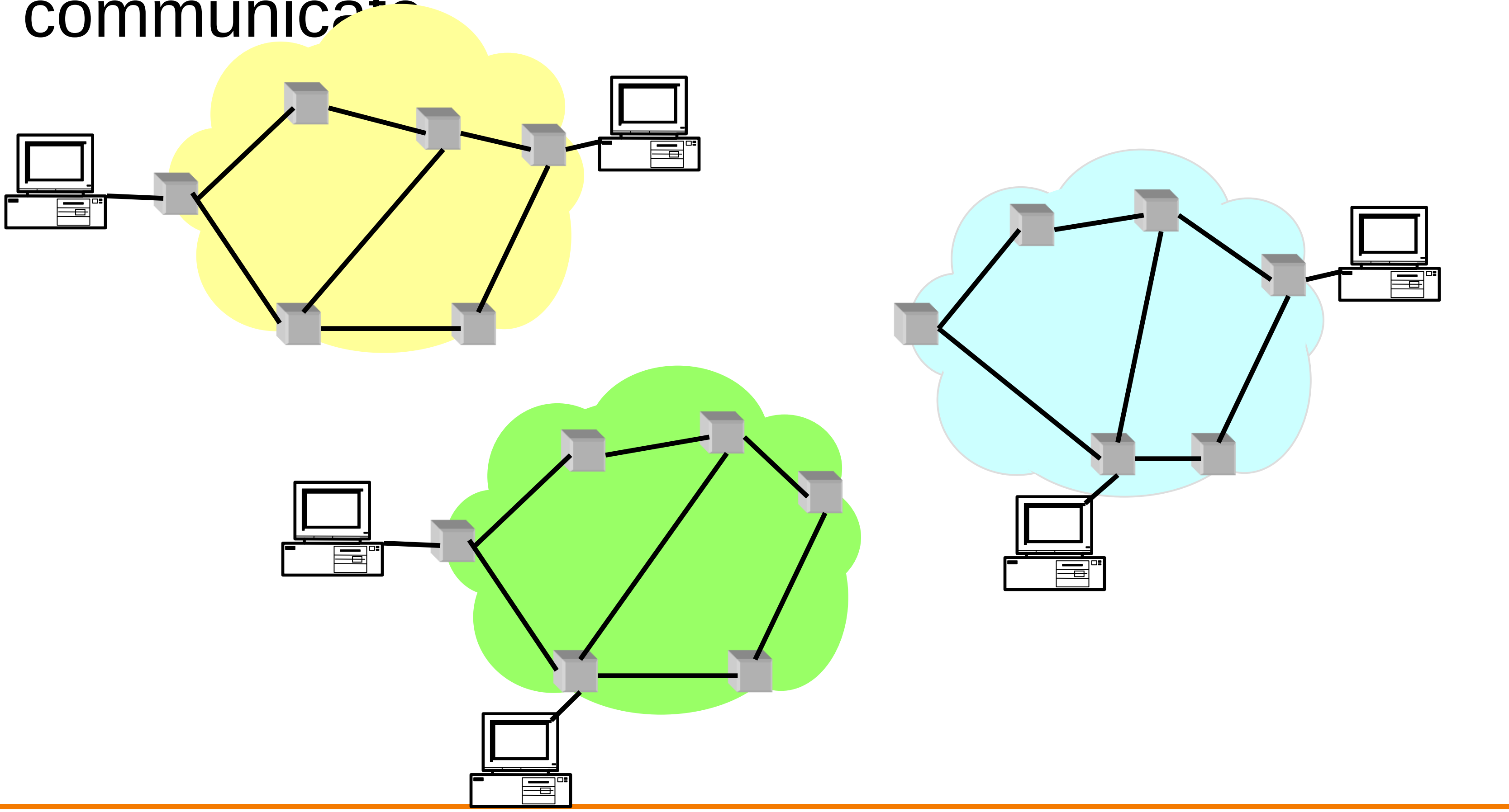
1972 Email invented

1972 Telnet introduced

1972 Kahn advocates Open Architecture networking

The Problem

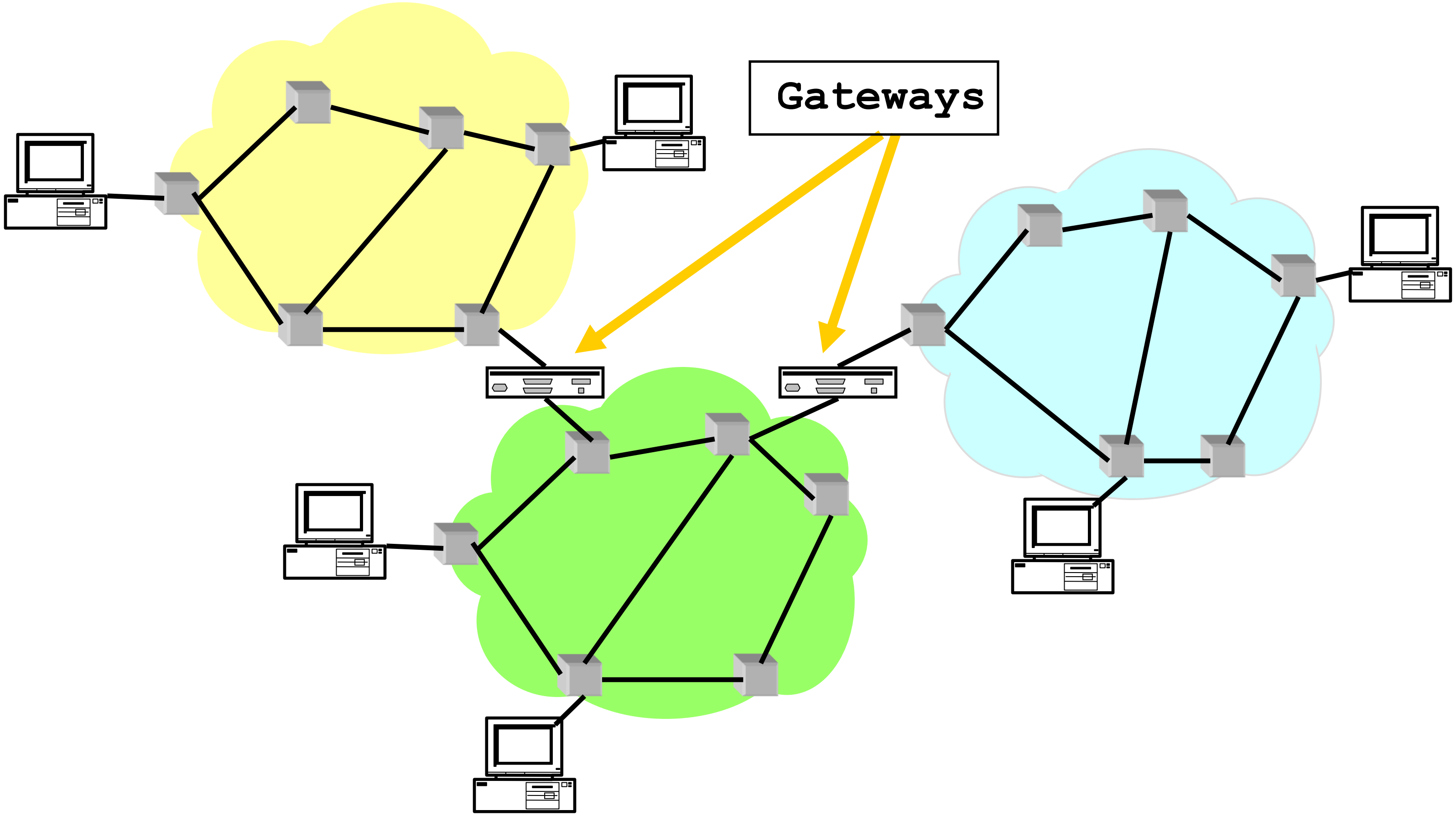
- Many different packet-switching networks
- Only nodes on the same network could communicate



Kahn’s Rules for Interconnection

- Each network is independent and must not be required to change (why?)
- Best-effort communication (why?)
- Boxes (routers) connect networks
- No global control at operations level (why?)

Solution



Kahn's vision

- Kahn imagined there would be only a few networks (~20) and thus only a few routers
- He was wrong
  - Why?
- Imagined gateways would "translate" between networks
  - We think of it as all routers supporting IP

Timeline continued….

1973 FTP introduced

1974 Cerf and Kahn paper on TCP/IP

1980 TCP/IP adopted as defense standard

1983 Global NCP to TCP/IP flag day

198x XNS, DECbit, and other protocols

1984 Janet (British research network)

1985 NSFnet (picks TCP/IP)

198x Internet meltdowns due to congestion

1986 Van Jacobson saves the Internet (BSD TCP)

Unsung hero of Internet: David D. Clark

- Chief Architect 1981-1988
- Great consistency of vision
- Kept the Internet true to its basic design principles
- Authored what became known as the End-to-end principle (next lecture)
- Conceives <u>and</u> articulates architectural concepts
  - Read his “Active Networking and End-To-End Arguments”
- Perhaps the only “irreplaceable” Internet pioneer

Timeline continued…

1988 Deering and Cheriton propose multicast

1989 Birth of the web….Tim Berners-Lee

Why did it take physicist to invent web?

- Physicists are the smartest people in the world?
- Computer scientists were trying to invent nirvana
  - Well, actually Xanadu (Ted Nelson)
  - More generally, CS researchers focused on hyptertext
- Again, users didn't need what we wanted to invent
  - Think about it: a paper on the web design would have been rejected by every CS conference and journal
- In general, the CS research community is great at solving well-defined problems, but terrible at guessing what users will actually use
  - "Academics get paid for being clever, not for being right." …Don Norman

Timeline continued…..

1993 Search engines invented (Excite)

199x ATM rises and falls (as internetworking layer)

199x QoS rises and falls

1994 Internet goes commercial

1998 IPv6 specification

1998 Google reinvents search

200x The Internet boom and bust

2012 **EE122 enrollment suggests boom is back!**

~80 in 2010 to ~200 in 2011 to ~340 in 2012